Practical implementation of promise theory in CFEngine

Mikhail Gusarov
CFEngine AS
<mikhail.gusarov@cfengine.com>
Contents

- Promise theory
- "reports"
- "processes"
- "packages"
- "variables"
- Summary
Promise theory

«Agent A promises to agent B to fulfill C»
Software Implementation

- Convergency
  - Promise essence is the «desired state»
- Embracing errors
  - Promises may be kept occasionally
- Autonomy
  - Unable to rely on another agents
Real-world example: e-mail

- Convergency
  - Delivery to address

- Embracing errors
  - 4xx, 5xx errors
  - Multiple MXes
  - Mail queues

- Autonomy
  - Independent mail servers
  - Local routing decisions
Reports

body common control {
    bundlesequence => {"mybundle"};
}
bundle agent mybundle {
    reports:
        linux::
            "Hello world!";
}
Reports

body common control {
  bundlesequence => {"mybundle"};
}

bundle agent mybundle {
  reports:
    linux::
      "Hello world!";
}
bundle agent mybundle {
    reports:
        linux::
            "Hello world!";
}
Reports

text

bundle agent mybundle {
  reports:
  linux::
    "Hello world!";
}
bundle agent mybundle {
reports:
    linux::
        "Hello world!";
}
Reports

bundle agent mybundle {
  reports:
    linux::
      "Hello world!";
}

Reports (translated)

```python
def mybundle() {
    if "linux" in global_contexts {
        report("Hello world!");
    }
}
mybundle()
```
Reports (run)

$ cf-agent
R: Hello world!
$
$
Reports (run)

$ cf-agent
R: Hello world!
$ cf-agent
$
Reports (run)

$ cf-agent
R: Hello world!

$ cf-agent

$ cf-agent

$
Reports

bundle agent mybundle {
  reports:
    linux::
      "Hello world!";
      "That's surprising";
}
$ cf-agent
R: Hello world!
$ cf-agent
$ cf-agent
$ cf-agent
R: That's surprising
$
Reports (run)

$ cf-agent
R: Hello world!
$ cf-agent
$ cf-agent
$ cf-agent
R: That's surprising
$ cf-agent
R: Hello world!
$
```python
def mybundle() {
    promises.add(
        report("Hello world!",
            activate = λ:
            "linux" in
            global_contexts))
}
mybundle()
run_all_relevant_promises()
```
«Ensure that specified message is reported to sysadmin as soon as possible, but no more than once per 5 minutes to the sysadmin using the specified reporting mechanisms»
«Ensure that specified message is reported to sysadmin as soon as possible, but no more than once per 5 minutes to the sysadmin using the specified reporting mechanisms»
Embracing errors

«Ensure that specified message is reported to sysadmin as soon as possible, but no more than once per 5 minutes to the sysadmin using the specified reporting mechanisms»
«Ensure that specified message is reported to sysadmin as soon as possible, but no more than once per 5 minutes to the sysadmin using the specified reporting mechanisms»
bundle agent myprocesses {
    processes:
        "nginx"
        match_range => irange("1","20");
        restart_class => "restart.nginx";
    commands:
        restart.nginx::
            "/etc/init.d/nginx start";
}
Processes

bundle agent myprocesses {
processes:
  "nginx"
  match_range => irange("1","20");
  restart_class => "restart_nginx";
commands:
  restart_nginx::
    "/etc/init.d/nginx start";
}
bundle agent myprocesses {
processes:
  "nginx"
  match_range => irange("1","inf");
  restart_class => "restart_nginx";
commands:
  restart_nginx::
    "/etc/init.d/nginx start";
}
def myprocesses() {
    nginx_cnt = count_proc("nginx")
    if nginx_cnt == 0 {
        promises.add(command(
            command("/etc/init.d/nginx start")))
    } elif nginx_cnt > 20  {
        Promises.add(kill("nginx",
            nginx_cnt - 20))
    }
}

myprocesses()
run_all_relevant_promises()
def myprocesses() {
    promises.add(
        processes("nginx",
            \ count: if count == 0 {
                define_context("restart_nginx")
            })))
    promises.add(
        commands(
            "/etc/init.d/nginx start",
            activate_if = \: 
                "restart_nginx" in global_contexts))
    }
myprocesses()
run_all_relevant_promises()
«Ensure that amount of specified processes is kept in specified range, killing extra processes and communicating the need to restart if necessary»
«Ensure that amount of specified processes is kept **in specified range**, killing **extra processes** and communicating the need to **restart** if necessary»
Embracing errors

«Ensure that amount of specified processes is kept in specified range, killing extra processes and communicating the need to restart if necessary»
«Ensure that amount of specified processes is kept in specified range, killing extra processes and communicating the need to restart if necessary»
Packages

bundle agent mypackages {
  packages:
    "libapache2-mod-wsgi"
    package_method => apt,
    package_policy => "addupdate",
    package_version => "3.3-4",
    package_select => ">=";
}
bundle agent mypackages {
  packages:
    "libapache2-mod-wsgi"
  package_method => apt,
  package_policy => "addupdate",
  package_version => "3.3-4",
  package_select => ">="
}
Packages (translated)

(Implementation is too large to fit in the marginslide)
«Ensure that specified package is installed or not installed according to the constraints specified. Do this by instructing local package manager to add, upgrade or remove package»
«Ensure that specified package is installed or not installed according to the constraints specified. Do this by instructing local package manager to add, upgrade or remove package»
Embracing errors

«Ensure that specified package is installed or not installed according to the constraints specified. Do this by instructing local package manager to add, upgrade or remove package»
«Ensure that specified package is installed or not installed according to the constraints specified. Do this by instructing local package manager to add, upgrade or remove package»
Variables

bundle agent myvars {
  vars:
  "a" string =>
    "My $\{\text{string}\}";
}
«Maintain in-agent binding of left-side name to the value of right-side expression»
Convergent, autonomous

«Maintain in-agent binding of left-side name to the value of right-side expression»
bundle agent myvars {
  vars:
  "myvar" string => execresult(
    "/usr/bin/testparm
    /etc/smb/smb.conf",
    "noshell"));
}
Summary

- CFEngine building blocks are promises
  - Higher-level
  - Convergent
  - Error-embracing
  - Autonomous

- Not the things from conventional languages
Questions?
DSL vs. eDSL

- **Pro eDSL:**
  - Familiar host language

- **Contra eDSL:**
  - Radically different building blocks
  - Different control structure and evaluation

→ DSL